1986

Finding Maximum Cliques on Circular-Arc Graphs

Alberto Apostolico

Susanne E. Hambrusch
Purdue University, seh@cs.purdue.edu

Report Number:
86-643

Apostolico, Alberto and Hambrusch, Susanne E., "Finding Maximum Cliques on Circular-Arc Graphs"
(1986). *Department of Computer Science Technical Reports*. Paper 559.
<https://docs.lib.purdue.edu/cstech/559>

FINDING MAXIMUM CLIQUES ON
CIRCULAR-ARC GRAPHS

Alberto Apostolico
Susanne Hambrusch

CSD-TR-643
December 1986

Finding Maximum Cliques on Circular-Arc Graphs

Alberto Apostolico and Susanne Hambrusch*

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA.

CSD-TR-643
December 1986

ABSTRACT

An algorithm is presented that, given the intersection model S of a circular-arc graph G with n vertices and m edges, finds a maximum-sized clique of G in $O(n^2 \log \log n)$ time. The previously best time bound for this problem is $O(n^2 \log n + mn)$.

---

* This work was supported by the Office of Naval Research under Contracts N00014-84-K-0502 and N00014-86-K-0689, and by the National Science Foundation under Grant DMC-84-13496.

1. Introduction

Let $G=(V,E)$ be an undirected graph with $|V|=n$ vertices and $|E|=m$ edges. Graph G is a *circular-arc graph* if its vertices can be put in a one-to-one correspondence with the elements of some set S of arcs on the unit circle, in such a way that two vertices are adjacent in G if and only if the two corresponding arcs in S have a non-empty intersection [Go]. We call S the *intersection model* of G . Circular arc graphs enter diverse applications, for which we refer to [Go, H].

Let an arbitrary clockwise system of abscissae be defined on the unit circle. Then the i -th arc in S is identified by the pair (a_i, b_i) , where a_i is the abscissa of the counterclockwise end of the arc and b_i is the abscissa of the clockwise end. We say that arc segment (l, r) (where l is the counterclockwise end and r is the clockwise end) contains position x if either $l < x < r$, or $l > r$ and $(x > l$ or $x < r)$. Arc i contains arc j if (a_i, b_i) contains both position a_j and b_j . Arcs i and j intersect if either one of the two arcs is contained in the other one, or the two arcs overlap. A subset C of S is a *clique* if each arc in C intersects all other arcs in C . A clique C is *maximum* if its cardinality is not smaller than that of any other clique of S .

In this paper we present an algorithm that, given the intersection model S of a graph G , finds a maximum clique of G in $O(n^2 \log \log n)$ time. The previously best time bound for this problem is $O(n^2 \log n + mn)$ [H]. Our solution uses some simple properties which we present in the following section. The main construction is then discussed in Section 3. In Section 4, we briefly point to some problems that remain open. For other NP-complete problems that are NP-complete in general, but can be solved efficiently on circular-arc graphs we refer the reader to [J].

2. Preliminaries

We assume without loss of generality that all arcs in S have length less than 1 and that no two arcs share a common endpoint.

Let C be a clique of S and let $i \in C$ be an arc which does not contain any other arc of C . Then we say that arc i is a *base* for C and we call C a *clique of base i* . Clearly, a maximum clique of S can be obtained by constructing, for every arc i , a clique of base i and maximum size, and then selecting one largest such clique. Let now C be a clique of base i and maximum size. The following two facts hold trivially for C .

Fact 1. All arcs of S that contain arc i are in C .

Fact 2. Every arc of S not contained in i , but having both endpoints in (a_i, b_i) is also in C .

Thus the arcs of C can be partitioned into three sets C_i^1 , C_i^2 , and C_i^3 , where

$$C_i^1 = \{j \mid \text{arc } j \text{ contains arc } i\}$$

$$C_i^2 = \{j \mid \text{arc } i \text{ does not contain arc } j, \text{ but both } a_j \text{ and } b_j \text{ are in } (a_i, b_i)\}$$

$$C_i^3 = \{j \mid \text{either } a_j \text{ or } b_j, \text{ but not both, are in } (a_i, b_i)\}$$

Note that, given base i , the arcs in C_i^1 and C_i^2 are uniquely determined, but the same is not true of C_i^3 . However, the arcs in C_i^3 must form a clique of maximum size among all cliques that contain only arcs with exactly one endpoint in (a_i, b_i) . Henceforth, we will use C_i^3 to denote *any* such maximum-sized clique.

3. Finding a Maximum Clique of Base i .

In this section we show that a maximum clique of base i can be constructed in $O(n \log \log n)$ time. The non-trivial part of our strategy is in its way of constructing set C_i^3 , since sets C_i^1 and C_i^2 can be trivially extracted from S in linear time.

Once C_i^1 and C_i^2 are removed from S , our algorithm forms set S_i , where

$$S_i = S - C_i^1 - C_i^2 - \{j \mid \text{arc } i \text{ contains arc } j\} - \{j \mid \text{arc } i \text{ and arc } j \text{ do not intersect}\}.$$

Thus set S_i contains all the arcs with exactly one endpoint in (a_i, b_i) . Every arc in S_i is a candidate for C_i^3 . An optimal selection of arcs can be performed conveniently on the *rollout* of the circle associated with arc i . This rollout is obtained by opening the circle at endpoint a_i , and then by mapping each circular arc of S_i into a suitable rectilinear segment. With reference to Figure 1, an arc j in S_i with a_j contained in (a_i, b_i) is mapped in the *A-segment* (a_j, b_j) , while an arc with b_j contained in (a_i, b_i) is mapped in the *B-segment* (b_j, a_j) (in a sense, a B-segment complements the circular arc associated with it).

Without loss of generality, let $b_i=0$ and $k = |S_i|$, and let $(l_1, r_1), \dots, (l_k, r_k)$ be the list of the segments of the rollout of i in order of increasing right endpoint (i.e., $l_j < 0 < r_j$ and $r_j = j$). We observe that the subset of S_i represented by the collection of all the A-segments in the rollout is a clique, and so is the union of this subset and $\{C_i^1 \cup C_i^2\}$. The same properties hold for the subset of S_i which is mapped into the set of B-segment. We say that A-segment x and B-segment y are

in conflict if $l_y < l_x < r_x < r_y$. In Figure 1, for example, A-segment 2 is in conflict with B-segments 3 and 6. Obviously, set C_i^3 cannot contain any pair of arcs such that the two corresponding segments are mutually conflicting.

The procedure TRADE described below performs an optimal selection of A-segments and B-segments in the rollout of i . Through a left-to-right scan of endpoints in $[1, k]$, TRADE constructs a set C3 containing the maximum number of non-conflicting segments of the rollout. We refer to C3 as a *maximum rollout clique of base i* . The procedure constructs then C_i^3 as the set of all arcs which correspond to segments in C3. By our discussion, taking the union of this set and $\{C_i^1 \cup C_i^2\}$ yields a maximum clique of base i .

TRADE initializes set C3 to contain all the A-segments of the rollout. Subsequently, the procedure may "match-up" A-segments with conflicting B-segments according to a criterion described shortly. These match-ups enable TRADE to recognize situations in which an equal number of A-segments can be "exchanged" for an equal number of B-segments without creating any conflicts. At the j -th iteration, segment (l_j, r_j) is handled as follows. If (l_j, r_j) is a B-segment not in conflict with any A-segment currently in C3, then this B-segment is added to C3. Otherwise, let p be the number of A-segments in C3 that (1) are in conflict with the B-segment (l_j, r_j) and (2) have not been matched-up with a conflicting B-segment. If $p = 1$, TRADE exchanges the conflicting A-segment with the current B-segment, and possibly performs other exchanges as well, as specified later. The overall operation yields a consistent (i.e., with no conflicting segments) set C3 having the same cardinality as the original set, and it may have the effect of reducing the number of A-segments that could be in conflict with B-segments in the future. If $p > 1$, TRADE cannot perform an exchange without decreasing the current cardinality of C3. However, such an exchange might prove profitable at some later stage, since the point could be reached where removing a certain number of A-segments would make room for a larger number of B-segments. In view of this possibility, the procedure matches up B-segment j with the conflicting and currently unmatched A-segment x having the smallest left endpoint. The matched pair (x, j) is then recorded in a list associated with the unmatched A-segment y such that $l_y > l_x$ and l_y is as small as possible.

The management of A-segments is quite simpler. Whenever an A-segment is met during the

scan, TRADE adds it to the list AVAIL. This list keeps track of all the already scanned A-segments in C3 that are yet to be matched. In addition, every A-segment y stored in AVAIL is provided with an initially empty list L_y . At each iteration for which y is kept in AVAIL, L_y will contain zero or more pairs representing currently unexchanged matches. AVAIL is used also to determine how many A-segments are in conflict with a given B-segment. A more formal description of TRADE is as follows.

Procedure TRADE

Input: The segments $(l_1, r_1), \dots, (l_k, r_k)$ of the rollout of i .

Output: A set C_i^3 .

begin

(1) Initialize: $C3 = \{j \mid \text{arc}(l_j, r_j) \text{ is an A-segment}\}; \text{AVAIL} = \emptyset$

(2) Scan:

for $j=1$ to k **do**

if (l_j, r_j) is an A-segment **then** $\text{AVAIL} = \text{AVAIL} \cup \{j\}; L_j = \emptyset$

else (* (l_j, r_j) is a B-segment *)

$\text{conflict\#} = |\{h \mid h \in \text{AVAIL} \text{ and } l_j < l_h\}|$

        (2.1) **if** $\text{conflict\#} = 0$ **then** $C3 = C3 \cup \{j\}$

        (2.2) **if** $\text{conflict\#} = 1$ **then** (* an exchange is performed *)

let x be the A-segment in AVAIL with $l_x > l_j$;

let the elements in list L_x be $(\alpha_1, \beta_1), \dots, (\alpha_s, \beta_s), s \geq 0$;

$\text{AVAIL} = \text{AVAIL} - \{x\}$;

$C3 = C3 \cup \{j, \beta_1, \dots, \beta_s\} - \{x, \alpha_1, \dots, \alpha_s\}$

        (2.3) **if** $\text{conflict\#} > 1$ **then** (* a match-up is recorded *)

let x be the A-segment in AVAIL with $l_x > j_j$ and l_x as small as possible;

let y be the A-segment in AVAIL with $l_y > l_x$ and l_y as small as possible;

$\text{AVAIL} = \text{AVAIL} - \{x\}$;

append list L_x together with the entry (x, j) to list L_y

endfor

(3) Output:

$C_i^3 = \{j \mid (l, r) \text{ is an A-segment of } C3, j \in S_i, \text{ and } l=a_j \text{ and } r=b_j\} \cup$
      $\{j \mid (l, r) \text{ is a B-segment of } C3, j \in S_i, \text{ and } l=b_j \text{ and } r=a_j\}$

end.

Observe that exchanging s A-segments for s B-segments as done in step (2.2) is not only at least as good as keeping those A-segments in the current version of C3. In fact the number of B-

segments that could be added to C_3 in the future cannot ever be decreased by the exchange, and may be actually increased. Before describing the implementation of TRADE, we prove that the segments in C_3 at the outset correspond to a maximum rollout clique of base i . This immediately implies that the corresponding set C_i^3 as generated in step (3) represents a maximum clique among the arcs in S_i .

Let $C_3^{(0)}$ be the set of all the A-segments in the rollout of i , and let $C_3^{(j)} \ 1 \leq j \leq k$ be the set of segments in C_3 at the end of the j -th iteration of TRADE. Clearly, $|C_3^{(0)}| \leq |C_3^{(1)}| \leq \dots \leq |C_3^{(k)}|$. The following lemma characterizes the sets $C_3^{(j)}$, ($0 \leq j \leq k$) and also establishes the basic invariants of TRADE.

Lemma 1. By the end of the j -th iteration of TRADE ($0 \leq j \leq k$), the following properties hold.

- (1) For every A-segment α such that $\alpha < j$ and $\alpha \notin C_3^{(j)}$, there exists a distinct B-segment β in $C_3^{(j)}$ such that $\beta \leq j$ and α and β are in conflict.
- (2) Every B-segment β such that $\beta < j$ and $\beta \notin C_3^{(j)}$ is in conflict with at least two A-segments in $C_3^{(j)}$.
- (3) For any set B of B-segments not in $C_3^{(j)}$ and having endpoints as in (2) above, there exists a set A of A-segments in $C_3^{(j)}$ such that $|A| > |B|$, and every segment in A is in conflict with at least one segment in B .

Proof: Property 1 follows from straightforward induction on j : the matched pairs traded in the course of all executions of step (2.2) up to the j -th iteration represent the relation of the claim.

As for Property 2, observe first that in order for some B-segment $\beta < j$ not to be included in $C_3^{(j)}$, the β -th iteration of TRADE must have resulted in case (2.3). This entails the existence of at least two A-segments in $C_3^{(\beta-1)}$ which were unmatched and also in conflict with β at the time of the β -th iteration. Let then $U = \{\alpha_1, \alpha_2, \dots, \alpha_s\}$ be the set, sorted in order of increasing left endpoint, of all the unmatched A-segments in conflict with β at the beginning of the β -th iteration. Step (2.3) of TRADE puts the pair (α_1, β) in the list L_{α_1} . Assume for a contradiction that the number of A-segments in U that are also in $C_3^{(j)}$ is less than two. Then, since $\alpha_1 \in C_3^{(j)}$, α_2 cannot be in $C_3^{(j)}$. By the control structure of TRADE, this implies that there is some B-segment β' , $\beta < \beta' \leq j$ such that either (Case 1) α_2 is exchanged with β' at the β' -th iteration, or else (Case 2)

α_2 is matched up with β' at that iteration and then exchanged with it at some later iteration j' , $\beta' < j' \leq j$. With reference to Case 2, observe that the pair (α_2, β') is created and added to some L -list precisely when β' is handled by TRADE, and the pair (α_1, β) is added to that same list at that point. It is easy to check that, from this moment on, any list containing the pair (α_2, β') must also contain (α_1, β) . Since lists are traded in an exhaustive fashion in case (2.2), any trade that exchanges α_2 with β' must also exchange α_1 with β . But this contradicts the hypothesis that $\beta \notin C3^{(j)}$. The same contradiction is easily derived for Case 1.

Property 3 is an easy consequence of the preceding two. Indeed, let $\beta_1, \dots, \beta_s$ be the segments in set B , with $r_{\beta_1} < \dots < r_{\beta_s}$. At the time TRADE handles β_q ($1 \leq q \leq s$) this B-segment is matched to a corresponding and distinct A-segment α_q . By Property 2, there must be another A-segment y in $C3^{(j)}$ which is distinct from any α_q and is also in conflict with β_s . Setting then $A = \{\alpha_1, \alpha_2, \dots, \alpha_s, y\}$ yields $|A| = s+1 > s = |B|$. $\square$

We now show that the set $C3 = C3^{(k)}$ is a maximum rollout clique of base i . Let $C3^*$ be a maximum rollout clique of base i that has as many arcs as possible in common with clique $C3$.

Lemma 2. $C3 = C3^*$.

Proof: Recall that the segments of the rollout of i are $(l_1, r_1), \dots, (l_k, r_k)$ with $r_j = j$. For $0 \leq j \leq k$, let RE_j (resp. RE_j^*) be the set of segments in $C3$ (resp. $C3^*$) having right endpoints not larger than j . Observe that $RE_0 = RE_0^* = \{i\}$. Let $p > 0$ be the smallest integer for which an "adversary" who knows the composition of set $C3^*$ claims that $RE_p \neq RE_p^*$. We will force this adversary in a contradiction, thus showing that $C3 = C3^*$. We distinguish two cases, depending on whether segment p is a B-segment or an A-segment.

Case 1. Segment p is a B-segment.

Assume first that $p \in C3$ but $p \notin C3^*$. It is easy to see then that p is not in conflict with any segment having right endpoint larger than p . Thus, whatever the remaining selections in $C3^*$, adding segment p to $C3^*$ yields a set of nonconflicting segments having larger size than $C3^*$, a contradiction. Assume now $p \notin C3$. By the second property of Lemma 1, there exist two A-segments in RE_{p-1} that are in conflict with segment p . Since both A-segments are also in $C3^*$ by hypothesis, $p \notin C3^*$ follows. In conclusion, p cannot be a B-segment, for otherwise we would have $RE_p = RE_p^*$.

Case 2. Segment p is an A-segment.

Assume first that $p \in C3$, $p \notin C3^*$. Then p must be in conflict with at least one B-segment of $C3^*$, otherwise it could be added consistently to this set, thus generating a contradiction. Let j_1 be a B-segment in $C3^*$ that is in conflict with p . Then $j_1 \notin C3$. By the second property of Lemma 1, there exists another A-segment in $C3$ that is in conflict with j_1 . Let p_1 be this A-segment. Obviously, $p_1 > p$ and $p_1 \notin C3^*$. The adversary is now offered to trade-in segment j_1 for the two segments p and p_1 . Clearly, such an exchange cannot be accepted without violating the hypothesis that $C3^*$ has maximum size. Thus, there must be a B-segment j_2 in $C3^*$ that is in conflict with either p , or p_1 , or both. But then A-segments p and p_1 in $C3$ and B-segments j_1 and j_2 not in $C3$, and Lemma 1 imply the existence of a third A-segment $p_2 \in C3$ such that p_2 is in conflict with j_1 , or j_2 , or both. Again, the adversary cannot accept to exchange the B-segments j_1 and j_2 for the A-segments p , p_1 and p_2 without falling in contradiction. Thus, $C3^*$ must exhibit a new B-segment j_3 that is in conflict with one or more among p , p_1 , p_2 . Continuing with this argument, the point must be reached where the adversary is forced to accept the trade at the time $C3^*$ runs out of new conflicting B-segments. This leads to a contradiction since $C3^*$ is shown to be not of maximum size. Hence, if $p \in C3$, we have $p \in C3^*$ and $RE_p = RE_p^*$.

Consider now the case when $p \notin C3$ and $p \in C3^*$. Then there must be a B-segment in $C3$, say, j_1 , that is in conflict with p and is matched to p during the j_1 -th iteration of TRADE. The adversary is now offered to trade-in segment p in exchange for B-segment j_1 . Accepting the exchange would contradict the assumption that $C3^*$ agrees with $C3$ on a maximum number of segments. Thus $C3^*$ must contain another A-segment, say, p_1 , such that p_1 conflicts with j_1 and $p_1 > p$. Obviously, $p_1 \notin C3$ and, by the operation of TRADE, $C3$ contains a B-segment that was used at some point to replace p_1 . Let this B-segment be j_2 . Then again, we offer j_1 and j_2 in exchange for p and p_1 , thus forcing the adversary to exhibit a third A-segment. This argument continues until the adversary runs out of A-segments. In fact, $C3$ is always able to provide a new matching B-segment for any A-segment supplied for $C3^*$, in force of the first property of Lemma 1. Hence, if $p \notin C3$, we have $p \notin C3^*$ and $RE_p = RE_p^*$. $\square$

We now turn to the implementation of our algorithm. We can assume that the set S is stored in a circular array of $2n$ cells, with a convenient format. This can be obtained through a straight-

forward $O(n \log n)$ preprocessing, the details of which are omitted. The construction of the set of segments when handling base i can be performed trivially in linear time. An $O(k \log k)$ time implementation for TRADE is then easily obtained by using a bitvector to represent set C3 and a balanced tree for set AVAIL [AHU]. Every leaf x of this balanced tree corresponds to an A-segment and it has a linked list representing list L_x associated with it. In such an implementation $\text{conflict\#}$ can be computed in $O(\log k)$ time, step (2.1) requires constant time, and step (2.2) and (2.3) can each be done in $O(\log k)$ time. Thus the $O(k \log k)$ overall time follows. This time bound can be reduced to $O(k \log \log k)$ using the data structures described in [vE] and the references therein.

Theorem. A maximum rollout clique of base i can be constructed in $O(n \log \log n)$ time.

Proof: By the discussion of this section and Lemmas 1 and 2.

Corollary. Given n arcs $(a_1, b_1), \dots, (a_n, b_n)$ of a circular-arc graph G a maximum clique of G can be determined in $O(n^2 \log \log n)$ time.

4. Concluding Remarks.

A number of problems remain open. First, the techniques presented in this paper do not seem to extend straightforwardly to the more general problem of finding a clique of maximum weight for the intersection model of a weighted circular arc graph. By contrast, the $O(n^2 \log n + nm)$ technique in [H] applies to the weighted case as well. Thus the question is open of whether the weighted case can be dealt with more efficiently. For the case of unitary weights, it would be interesting to eliminate the logarithmic factor from the upper bound. Finally, lower bounds for both versions of the problem are still to be derived.

References

- [AHU] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.
- [Go] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [H] W.-L. Hsu, 'Maximum Weight Clique Algorithms for Circular-Arc Graphs and Circle Graphs', *SIAM J. on Computing*, pp 224-231, 1985.
- [J] D.S. Johnson, 'The NP-Completeness Column: an Ongoing Guide', *Journal of Algorithms*, Vol. 6, pp 434-451, 1985.
- [vE] P. van Emde Boas, 'Preserving Order in a Forests in less than Logarithmic Time and Linear Space', *Inform. Processing Letters*, pp 80-82, 1977.

(a) A circular-arc graph. Taking base $i=7$ yields $S_i = \{1,2,3,4,5,6\}$. Arc 8 is included into set S_7^1 , arc 9 is discarded when constructing a clique of base 7.

(b) the rollout of base $i=7$ consisting of A-segments 2, 4 and 5 and B-segments 1, 3, and 6 (shown with dashed vertical lines and a heavy horizontal line)

FIGURE 1